Mikroprocesor 8085 jest udoskonaloną wersją 8080. Architekturą i możliwościami nie dorównuje jednak Z80 i z tego względu jest — na świecie — znacznie rzadziej stosowany. Rozszyfrowanie dodatkowych, nie publikowanych przez producenta rozkazów zwiększa nieco efektywność programowania, ale nowej gwiazdy niestety nie kreuje. W Polsce „specjalizujemy się" w układach firmy INTEL i choćby były gorsze i droższe — jakaś magiczna siła zamienia skąpe ilości dolarów na produkty tej firmy lub ich odpowiedniki. Znając zaś prawidłowości naszej gospodarki i chcąc uniknąć iluzji na łamach mikroKLANU, nie zamierzamy walczyć z tą tendencją. Stąd też wynika przeświadczenie, że informacja o odkrytych (bynajmniej nie w Polsce) nowych rozkazach dla mikroprocesora 8085 będzie przydatna dla wielu mikroKLANOWCÓW.

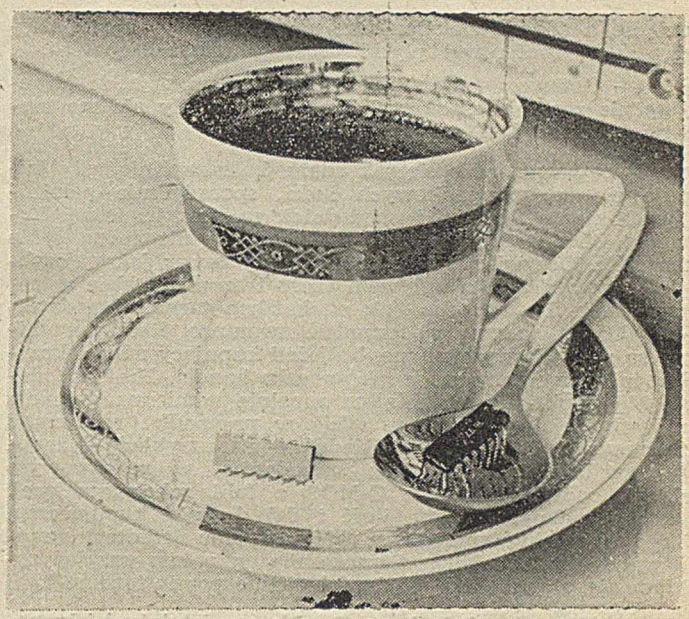
Nieznane rozkazy mikroprocesora 8085

Wśród 256 możliwych kodów rozkazów mikroprocesora 8080, dwanaście to kody niewykorzystane nie występujące w oficjalnych materiałach firmowych. Mikroprocesor 8085 wykorzystuje dodatkowo dwa z tych kodów dla rozkazów RIM i SIM (opisane w materiałach firmowych i podręcznikach firmy INTEL).

Z niewyjaśnionych przyczyn znaczenie pozostałych dziesięciu kodów nie zostało dotąd opublikowane przez producenta. Na podstawie opracowanych testów dla mikroprocesorów w wersji 8085 i 8085A odkryto znaczenie tych dziesięciu kodów rozkazów, a także znaczenie dwóch bitów w rejestrze stanu. Siedem spośród tych dziesięciu rozkazów dotyczy operacji na danych 16-bitowych, a trzy pozostałe wykorzystują „nowe" bity stanu. Proponowane mnemoniki dla nowych rozkazów są zgodne z regułami stosowanymi przez firmę INTEL. W tabeli zestawiono kody nowych rozkazów, ich mnemoniki oraz skrócony opis działania.

Nieznane operacje mikroprocesora Intel 8085

Kod operacji (hex.)	Mnemonik	Zmienia bity stanu	Liczba cykli		Skrócony opis działania
			maszynowych	zegarowych	
08	DSUB	Z, S, P, CY, AC, X5, V	3	10	(H)(L) = (H)(L) − (B)(C)
10	ARHL	CY	2	7	(H7) = (H7), (Hn−1) = = (Hn), (L7) = (H0), (Ln−1) = (Ln), (CY) = = (L0)
18	RDEL	CY, V	3	10	(Dn+1) = (Dn), (D0) = = (E7), (CY) = (D7), (En+1) = (En), (E0) = (CY)
28	LDHI	–	3	10	(D)(E) = (H)(L) + (bajt 2)
38	LDSI	–	3	10	(D)(E) = (SPH)(SPL) + + (bajt 2)
CB	RSTV	–	1/3	6/12	(V = 1) → (SP) −1 = (PCH), ((SP)−2) = (PCL), (SP) = (SP)−2, (PC) = = 40H
D9	SHLX	–	3	10	((D)(E)) = (L), ((D)(E)+1) = (H)
DD	JNX5	–	2/3	7/10	(X5 = 0) → (PC) = = (bajt 3)(bajt 2)
ED	LHLX	–	3	10	(L) = ((D)(E)) (H) = ((D)(E)+1))
FD	JX5	–	2/3	7/10	(X5 = 1) → (PC) = = (bajt 3)(bajt 2)



Człowiek z Perskiego

µK: — Jak to się stało, że sprzedajesz na Perskim Jarmarku akurat części elektroniczne?

C: — Dość prozaicznie. Niektórych elementów miałem więcej, innych potrzebowałem... I tak zacząłem przychodzić na Perskł. Ku mojemu zdziwieniu spotkałem tam wielu znajomych z Uczelni.

µK: — Z Uczelni? Jesteś profesjonalistą?

C: — Tak, skończyłem Wydział Elektroniki. Ale studia nie dają wiedzy praktycznej. Na Perskim przez pół roku zdobyłem chyba więcej wiadomości niż przez pięć lat Politechniki. Głównie dlatego, że zacząłem coś robić samemu i mogłem to skonfrontować z innymi praktykami.

µK: — Co daje Ci Perski? Dużo zarabiasz? Jeździsz własnym samochodem w odróżnieniu od innych absolwentów?
C: Nie, takich pieniędzy na tym nie robię, samochodu nie mam. No, stać mnie czasem na taksówkę czy filiżankę kawy, ale chyba nie na tym polega luksusowe życie? Handlu na dużą skalę nie prowadzę, bo mnie to nie bawi; zresztą chyba nie potrafię... W Perski po prostu wsiąkłem.

µK: — Zaobserwowałem tam niecodzienną atmosferę. Nieznajomi wymieniają doświadczenia jak bliscy koledzy. Sprzedający zupełnie nie przypominają sklepowej ekspedientki. Rozmawiają, radzą... Widziałem — na przykład — jak doradzano kupującemu, oferując mu tańsze elementy niż początkowo chciał zakupić.

C: — To jest logiczne! Człowiek zajmuje się tym, na czym się zna. Praktycznie wszyscy handlujący na Perskim znają się dokładnie na tym, co sprzedają, bo prawie każdy z nas jest hobbystą. Oczywiście, jak w każdym środowisku, tak i na Perskim jest pewna grupa ludzi, którzy zajmują się tym wyłącznie dla handlu — mają tzw. dojścia i mogą na tym zarobić. To jest 10, może 20% ogółu. Wyspecjalizowanych hobbystów jest jednak znacznie więcej, co widać zresztą po asortymencie: ktoś ma tylko „analogówkę", inny „cyfrówkę", a ostatnio pojawili się specjaliści od mikroprocesorów. Co więcej — oferowane są też półprodukty i gotowe urządzenia własnego wyrobu: wzma-

Człowiek z Perskiego

cniacze, zasilacze, elektroniczne pozytywki czy zegary. Niedawno ktoś oferował płytki prostego mikrokomputera.

µK: — Gotowe urządzenia... Jaka jest gwarancja, że kupujący nie zostanie oszukany? Przecież z reguły nie ma możliwości sprawdzenia towaru na miejscu.

C: — Zapewne może się tak zdarzyć. Szczególnie, gdy kupuje się od przypadkowych ludzi, sporadycznie pokazujących się na Perskim i na dodatek za okazyjną cenę. Bywalcy Perskiego na taką kompromitację nie mogą sobie pozwolić. Oczywiście, należy zakupiony towar jak najszybciej sprawdzić, bo z upływem czasu transakcja idzie w zapomnienie. Na Perskim, jak na każdym targu, niezadowolony klient ma p r a w o urządzić awanturę. Rzeczy wadliwe są z reguły wymieniane, bo jeżeli kupujący narobi krzyku, to sprzedający ma z głowy cały handel. Nikt od niego nie kupi... Tu ma miejsce zupełnie odmienna sytuacja niż w oficjalnym handlu, gdzie w przypadku reklamacji zawsze winny jest klient. Kiedyś — na przykład — zaobserwowałem taki kwiatek: sprzedawca, w państwowym sklepie, odwinął układy C-MOS z folii zabezpieczającej, odliczył palcami żądaną ilość i dał do ręki kupującemu. Trudno tu winić sprzedawcę, który pewnie przedtem sprzedawał ziemniaki. Niby dlaczego ma on potem przyjąć reklamację, skoro układy były fabrycznie przetestowane [1]).

µK: — Wracając jednak na Perski: czy istnieje ścisły podział na kupujących i sprzedających?

C: — Często zdarzają się wymiany między handlarzami, ten potrzebuje tego, tamten czegoś innego. To są przecież ludzie, którzy się elektroniką zajmują. Dlatego też nie warto oszukiwać. Grozi to utratą wiarygodności i kontaktów. Kiedyś zdarzyła mi sią taka sytuacja: na 80 wzmacniaczy operacyjnych — 20 to były tylko wyprowadzenia zalane żywicą. Oczywiście nabywca wrócił z reklamacją, lecz ja nie miałem już innych na wymianę. Pomógł mi sąsiad, który trzy stoły dalej sprzedawał takie same wzmacniacze, ale o 20 zł drożej. Specjalnie dla mnie obniżył cenę, tak aby mój klient odszedł usatysfakcjonowany. A wracając jeszcze do podziału na dwie strony lady... Był taki moment, że dwóch chłopaków z technikum regularnie przez kilka tygodni zamęczało mnie pytaniami o obudowę zasilaczy: który „scalak" lepszy, jakie stosować oporniki, itd. Teraz sprzedają na Perskim zasilacze.

µK: — Czy nie wylania nam się tutaj zbyt sielankowy obraz Perskiego?

C: — Należy chyba zacząć od stwierdzenia, że oficjalne ceny półprzewodników są absurdalne w porównaniu z płacami. Jeśli liczyć nawet po 700 zł za dolar, a to i tak układy zachodnie są często tańsze (nie mówię tu oczywiście o cenach ze sklepu w „Intraco" [2])! Na Perskim zdarzają się niestety towary kupione w sklepach czy w BOMIS-ie — ja to osobiście potępiam jako zwyczajną spekulację. Są też rzemieślnicy montujący telewizory z niepełnowartościowych podzespołów (czasem o lepszej jakości niż fabryczne). Tak czy inaczej — Perski to bazar, po którym trzeba nauczyć się poruszać.

µK: — Jakie masz rady dla początkujących?

C: — Trzeba przyjść, obejrzeć. Potem drugi raz, trzeci. Porozmawiać kto co ma. Unikać okazji i ludzi sprzedających raptem trzy elementy. Trzeba po prostu poznać dziedzinę, w której chce się coś robić...

µK: — Przewija się ciągle w tej rozmowie edukacyjna rola Perskiego. Może widzisz w tym jakąś misję?

---

[1]) Dotknięcie palcem końcówki układu C-MOS z reguły powoduje jego zniszczenie przez ładunek elektrostatyczny — przyp. AJP

[2]) W budynku „Intraco" w Warszawie ma swój sklep firma UNITEX, sprzedająca za waluty wymienialne podzespoły i urządzenia produkowane na Zachodzie — przyp. AJP

DSUB — odejmowanie dwóch liczb 16-bitowych zawartych w rejestrach HL i BC; komplementarne z instrukcją DAD B

ARHL — przesunięcie arytmetyczne w prawo o jeden bit zawartości pary rejestrów HL poprzez bit przeniesienia, z powieleniem najstarszego bitu w rejestrze H

RDEL — rozszerzona na liczbę 16-bitową operacja przesunięcia cyklicznego w lewo o 1 bit (poprzez bit przeniesienia) zawartości pary rejestrów DE; analogiczna do operującej na liczbach 8-bitowych instrukcji RAL

LDHI — umożliwia realizację adresowania indeksowego poza 256-bajtową granicę strony; ładuje do DE zawartość HL powiększoną o podany bajt

LDSI — nadaje się w szczególności do operacji na stosie w różnych podprogramach; ładuje do DE wskaźnik stosu powiększony o podany bajt

RSTV — działa jako warunkowy rozkaz RESTART od adresu 40H w przypadku, gdy bit V w rejestrze stanu jest ustawiony na 1; w przeciwnym razie działa jak instrukcja NOP; bit V wskazuje na przepełnienie w trakcie wykonywania operacji na liczbach przedstawionych w uzupełnieniu dwójkowym

SHLX — zapamiętanie zawartości pary rejestrów HL w komórce o adresie wskazanym przez parę rejestrów DE

JX5, JNX5 — reagują na bit X5 analogicznie jak JC i JNC na bit przeniesienia CY i mogą być uważane jako warunkowe rozkazy skoku; bit X5 nie odpowiada żadnemu ze znanych standardowych bitów stanu, lecz określony jest przez swoją pozycję w rejestrze stanu — znajduje on zastosowanie jako wskaźnik przepełnienia dla liczb bez znaku w operacjach typu INX i DCX, w szczególności przy przejściu z FFFFH na 0000 (INX) lub z 0000 na FFFFH (DCX)

LHLX — ładowanie pary rejestrów HL zawartości komórki o adresie wskazanym przez parę rejestrów DE.

Wykorzystanie „nowych" rozkazów w programach napisanych w języku maszynowym następuje analogicznie do „starych" — w kolejnych komórkach pamięci umieszczony jest kod operacji i wymagana liczba bajtów z danymi. W przypadku współpracy z programem ASSEMBLER, który nie rozpoznaje użytych kodów mnemonicznych, najlepiej jest zdefiniować odpowiednie makroinstrukcje, które ustawiałyby kod nowego rozkazu i ewentualne argumenty jako bajty danych.

Nowe rozkazy nadają się w szczególności do wykorzystania w podprogramach arytmetycznych. Przykładowo — dzielenie liczby 16-bitowej przez liczbe 16-bitową z użyciem rozkazów RDEL i DSUB trwa średnio ok. 368 µs a mnożenie 278 µs. Programy takie są około dwukrotnie szybsze niż w przypadku nie stosowania tych nowych rozkazów.

Przykładowe programy pokazują wykorzystanie nowych rozkazów.

Przykład 1

```
; MUL 16- MNOŻENIE DWÓCH DODATNICH LICZB 16-BITOWYCH
; WYWOŁANIE : CALL MUL16
; PARAMETRY : BC - MNOŻNA (MSB, LSB)
;             DE - MNOŻNIK (MSB, LSB)
; WYNIK : 32 BITY W REJESTRACH DE (MSB), HL (LSB)
MUL16:  LXI   H,0       ; WYZEROWANIE PARY HL
        MVI   A,17      ; INICJOWANIE LICZNIKA BITÓW
MULA:   DCR   A         ; MNOŻENIE SKONCZONE?
        RZ              ; TAK, POWROT Z PODPROGRAMU
        DAD   H         ; WYNIK JEDEN BIT W LEWO
        RDEL            ; NOWY ROZKAZ - DE 1 BIT W LEWO
        JNC   MULA      ; BIT MNOŻNIKA = 1?
        DAD   B         ; TAK, DODAJ MNOZNA
        JNC   MULA      ; NIE MA PRZEPELNIENIA,
                        ; NASTĘPNY BIT
        INX   D         ; PRZEPELNIENIE Z HL DO DE
        JMP   MULA      ; NASTEPNY BIT
```

Przykład 2

```
; DIV32 – DZIELENIE LICZB ZNAKOWANYCH DODATNICH
; WYWOLANIE : CALL DIV32
; PARAMETRY : DE (MSB), HL (LSB) – DZIELNA 32-BITY (MSBit = 0)
;             BC (MSB) – DZIELNIK 16-BITOW (MSBit = 0)
; WYNIKI    : DE (MSB, LSB) – ILORAZ – 16-BITOW
;             : HL (MSB, LSB) – RESZTA
DIV32: XCHG                 ; WYMIANA ZAWARTOSCI
                            ; REJESTRÓW DE I HL
; BADANIE WYSTAPIENIA NADMIARU PRZY DZIELENIU
       DSUB                 ; CZY WYSTAPI NADMIAR?
       JNC   DIVOV          ; TAK, SKOCZ DO PROGRAMU
                            ; OBSŁUGI
; NIE WYSTAPI NADMIAR

       DAD   B      ; ODTWORZENIE MSB DZIELNEJ
       MVI   A,17   ; INICJOWANIE LICZNIKA BITÓW
DIVA:  DCR   A      ; DZIELENIE SKONCZONE?
       RZ           ; TAK, POWROT Z PODPROGRAMU
       DAD   H      ; PRZESUN MSB DZIELNEJ O 1 BIT W LEWO
       ANI   0FFH   ; USTAW CY = 0
       RDEL         ; PRZESUN LSB DZIELNEJ O 1 BIT W LEWO
                    ; (E0 = 0)
       JNC   DIVE   ; D7 = 1?
       INR   L      ; TAK, POWIEKSZ MSB DZIELNEJ
DIVE:  DSUB         ; UTWORZ RESZTE CZESCIOWA
       JC    DIVD   ; WYNIK ODEJMOWANIA UJEMNY?
       INR   E      ; NIE, TWORZ ILORAZ (E0 = 1)
       JMP   DIVA   ; NASTEPNY BIT
DIVD:  DAD   B      ; TAK, ODTWORZ RESZTE CZESCIOWA
       JMP   DIVA   ; NASTEPNY BIT
; PROGRAM OBSŁUGI NADMIARU PRZY DZIELENIU
DIVOV: .....
```

LITERATURA

[1] Dehnhardt W., Sorensen V. M.: Unbekannte 8085 Instruktionen. Elektronik, nr 15/1978, s. 66
[2] Dehnhardt W., Sorensen V. M.: Unspecified 8085 op codes enhance programming. Electronics, January 18, 1979, s. 144—145
[3] MCS-80 User's Manual, Intel, October 1977.

ZBIGNIEW BARBAŚ
Ośrodek Informatyki
Politechnika Poznańska

CENY ● CENY ● CENY ● CENY ● CENY

grudzień 1983

Podzespoły

USA ($):

Z80 — 4, Z80A — 5, Z80B — 10; 8080 — 3, 8085 — 5, 8085A-2 — 7, 8086 — 25; 6800 — 3, 6809E — 15; 68000 — 50; 6502 — 5, 6502A — 7, 6502B — 10; Z8001 — 45, Z8002 — 35; 8 × 4116 (120 ns) — 30, 8 × 4116 (200 ns) — 13; 4164 (150 ns) — 7, 4164 (200 ns) — 6;2716 (350 ns) — 6, 2716 (450 ns) — 4; 2732 (200 ns) — 12, 2732 (250 ns) — 9, 2732 (450 ns) — 5, 2764 (200 ns) — 25, 2764 (450 ns) — 10

Polska — UNITEX ($):

Z80A — 10,2, Z80A PIO — 11,4, Z80 PIO — 9,8, Z80A CTC — 11,4, Z80 CTC — 9,8, Z80A DMA — 33,6, Z80 DMA — 27,2, Z80A SIO — 37, Z80 SIO/2 — 28,5, Z80A DART — 27,2, Z80 DART — 23,3; 2716/4 — 10,2; 2114 — 5,3; 4116 — 4,3

RFN (DM):

Z80A — 8,5; 8080 — 9,5, 8085 — 10,3, 8086 — 65; 68000 — 220; 2716 — 9,8, 2732 — 16, 2764 — 20; 4116 (200 ns) — 2,9, 4164 (200 ns) — 14,8

Wielka Brytania (£)

Z80A — 2,43; 8080 — 2,5, 8085 — 3,5, 8086 — 18; 68000 — 78; 2716 — 2,25, 2732 — 3,5, 2764 — 5,9; 4116 — 0,75, 4164 — 3,5.

Człowiek z Perskiego

C: — Ludzie powinni uczyć się w szkole, na uczelni, z książki i czasopism fachowych. Perski to jedynie korepetycje. W niektórych dziedzinach znacznie poszerzone, wobec braku odpowiedniej literatury. Tu rządzą prawa rynku. Na przykład ci, którzy zajmują się mikroprocesorami, oferują zazwyczaj odbitki z katalogów, noty aplikacyjne, schematy... Rzeczy nieosiągalne w księgarniach. To może dowód absurdu, że handlarz z Perskiego lepiej wie czego ludzie potrzebują, niż wydawnictwa.

µK: — Czy nie mając — cioci w Ameryce, a mając jako zaplecze jedynie Perski, można zabrać się za budowę własnego mikrokomputera?

C: — Trzeba najpierw dysponować pewną wiedzą i doświadczeniem. Nie radzę nikomu zaczynać od dużych inwestycji. Ale prosty systemik — CPU, trochę RAM, EPROM, wyświetlacze na LED-ach — można skompletować za mniej niż 20 tys. Potem można go rozbudowywać. I na pewno warto — sądzę, że mikrokomputer o walorach ZX SPECTRUM można zbudować samemu za ok. 100 tys. zł.

µK: — A czy istnieje u nas rynek dla oprogramowania?

C: — Ciągle jeszcze w powijakach. Zbyt mało mikrokomputerów znajduje się w prywatnych rękach. W Polsce jest odwrotna sytuacja niż na Zachodzie — najdroższy jest sprzęt, a oprogramowanie można zawsze jakoś wykombinować. Ale zauważyłem, że kilku ludzi oferujących programy na ZX81 i SPECTRUM zaczęło już w miarę regularnie pojawiać się na Perskim. A zatem rynek taki zapewne niebawem powstanie, tym bardziej, że same układy można połączyć raptem na kilka sposobów — i na tym koniec. Dopiero oprogramowanie ożywia komputer.

µK: — Czy zdarza się, że przychodzi na Perski prywaciarz, który chce wykorzystać komputer „w interesie"?

C: — Mam właśnie przykład z ostatniej soboty. Do kolegi zwrócił się klient: „Chciałbym zautomatyzować pewien proces technologiczny, czy mógłby tym sterować mikroprocesor i jak by wyglądała cena takiego sterownika?

µK: — I dostał ofertę?

C: — Trudno było strzelać w ciemno, ale cena nie powinna przekroczyć 150—200 tys. zł. Klient stwierdził, że taka propozycja go interesuje. Wcześniej próbował w przedsiębiorstwach państwowych i firmach polonijnych. Ci pierwsi nie byli zainteresowani, ci drudzy ocenili inwestycję na co najmniej pół miliona.

µK: — Cyli na Perskim powstają zalążki nowych, bardziej efektywnych firm?

C: — Niezupełnie. W firmach polonijnych niekoniecznie siedzą tępaki. Gdyby kolega miał na utrzymaniu Urząd Podatkowy, księgowego i kilku kontrolerów, to chyba taniej by mu to nie wypadło.

µK: — A więc jest to w gruncie rzeczy działalność bez perspektyw!

C: — A czy Ty, jako redaktor mikroKLANU, masz jakieś wielkie perspektywy? Wszyscy jedziemy na tym samym wózku. Takie czasy.

µK: — Jeszcze jedno pytanie: czy masz do kogoś, o coś żal? Może o te długie godziny spędzone w upale lub na mrozie?

C: — Nie! Zrobiono bardzo dużo dając nam miejsce do spotkań. Nawet inwigilacja (wiadomo, że musi być) jest robiona raczej dyskretnie.

Tylko szkoda, że nowoczesne podzespoły to z importu, że informacja to kserokopie z zachodnich wydawnictw, a mikrokomputer kosztuje ponad dziesięć pensji, a nie jedną dziesiątą jak w krajach, gdzie na „pchlim targu" nie handluje się elektroniką.

Rozmawiał
ANDRZEJ J. PIOTROWSKI
Fot. A. Piastka

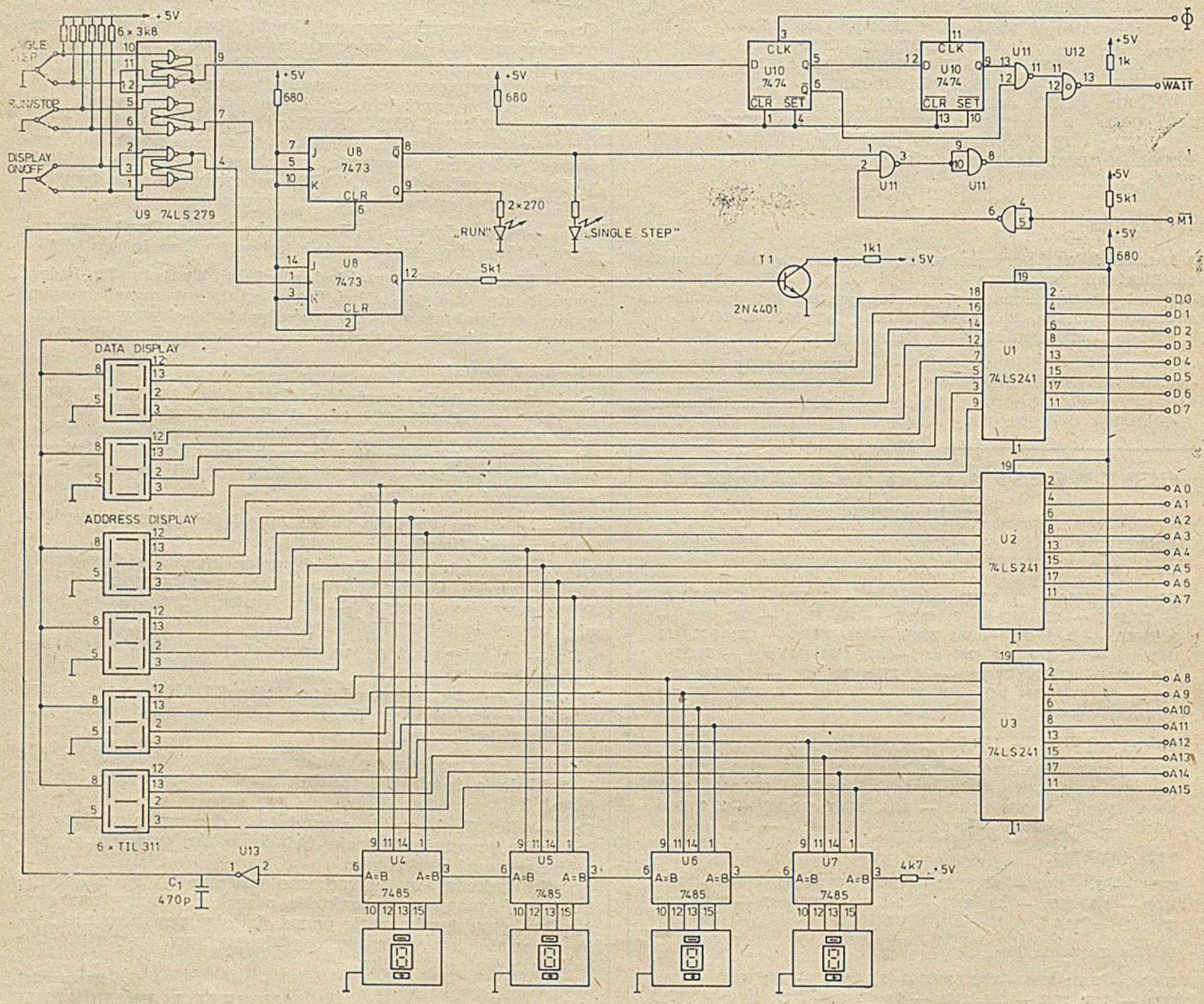
W pierwszym wydaniu MikroKLANU opisana została programowa realizacja pracy krokowej. Dla tych co wolą „plątać się w drutach" zamieszczamy rozwiązanie sprzętowe. Do jego niewątpliwych zalet należy możliwość „kroczenia przez program" zawarty w pamięci ROM. Dzięki temu można podpatrzyć niektóre tajemnice producentów mikrokomputerów lub zlokalizować błąd w tej pamięci. Układ można również wykorzystać przy uruchamianiu systemu nie wyposażonego w monitor ekranowy.

Koncepcja rozwiązania została zaczerpnięta z czasopisma „Elektronik" (kwiecień 1983), opisaliśmy jednak też sposób zaadaptowania jej do polskich warunków.

Układ pracy krokowej dla Z80

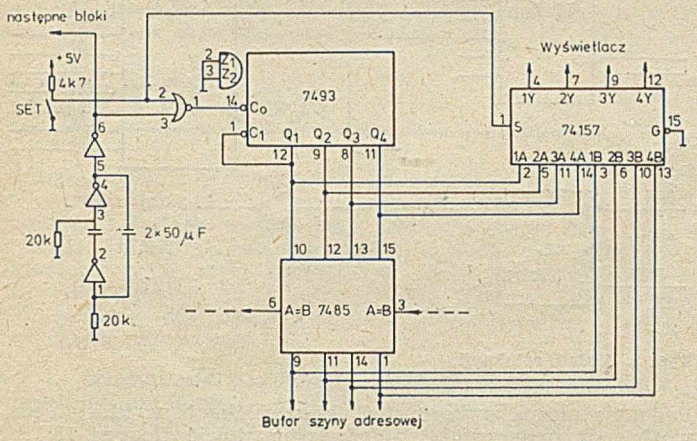
Prezentowany układ wstrzymuje pracę mikroprocesora po wykryciu zadanego kodu na szynie adresowej. Możliwe jest również ręczne wstrzymanie przez przyciśnięcie klawisza RUN/STOP. Po zatrzymaniu procesora można kontynuować program realizując krokowo kolejne rozkazy.

Adres, pod którym ma być wstrzymana realizacja programu, ustawiany jest za pomocą specjalnych przełączników rotacyjnych, wymuszających na czterech liniach wyjściowych kod dwójkowy ustawionej cyfry. Linie wyjściowe doprowadzone są do komparatorów (U4, U6 i U7), które dokonują porównania zadanych kodów z informacją znajdującą się na szynie adresowej. W przypadku stwierdzenia zgodności przez wszystkie cztery komparatory, zerowany jest przerzutnik U8. Kondensator C1 (rys. 1) służy do odfiltrowania krótkich impulsów (ang. glitch), mogących powstać przy zmianach wysterowania szyny adresowej. Sygnał z przerzutnika U8 poprzez bramki U11 i U12 podawany jest na linię $\overline{WAIT}$ i wymusza przejście procesora w stan oczekiwania. Przyciśnięcie klawisza „SINGLE STEP" zwalnia okresowo blokadę procesora, co umożliwia wykonanie jednego rozkazu. Przyciśnięcie klawisza RUN/STOP powoduje odblokowanie linii $\overline{WAIT}$. Powrót do pracy krokowej możliwy jest przez ponowne przyciśnięcie klawisza RUN/STOP.



Rys. 1. Schemat układu pracy krokowej dla Z80

Sygnały szyny danych i szyny adresowej są buforowane na wejściu układu (U1, U2 i U3), a następnie podawane na wyświetlacze 7-segmentowe o kodowaniu szesnastkowym (TIL-311).



Rys. 2. Układ elektronicznego przełącznika rotacyjnego

Niektóre układy zastosowane w prezentowanym rozwiązaniu są stosunkowo trudno osiągalne w kraju. Kosztem „elegancji" rozwiązania, a także zwiększenia poboru prądu ze źródła zasilania — można jednak dokonać adaptacji układu do polskich warunków. Bufory 74LS241 można z powodzeniem zastąpić polskimi odpowiednikami 8216, a nawet zwykłymi bramkami. Brak wyświetlaczy kodowanych szesnastkowo można ominąć, sterując zwyczajne wyświetlacze 7-segmentowe za pośrednictwem odpowiednio zaprogramowanych pamięci PROM o organizacji 32 × × 8 bitów. Tranzystor T1 należy dobrać w zależności od typu zastosowanych wyświetlaczy. Nietrudno też zastąpić układ 74LS279 stosując układy 74LS00 lub nawet 7400.

Nieco trudniej znaleźć substytut dla przełączników rotacyjnych. Można by zastosować niezależny przełącznik dla każdej linii (sterowanie binarne), ale jest to rozwiązanie wyjątkowo niedogodne dla użytkownika. Bardziej eleganckim, choć droższym, rozwiązaniem jest zastosowanie elektronicznych przełączników rotacyjnych. Na rysunku 2 przedstawiono schemat takiego rozwiązania (fragment dla jednej cyfry).

Zasadniczym elementem jest licznik binarny 7493 o równoległych wyjściach, które podawane są do komparatora oraz multipleksera szyny adresowej. Przyciśnięcie klawisza SET otwiera przejście dla impulsów zegara sterujących wejście licznika i równocześnie przełącza multiplekser adresów, tak aby na wyświetlacz podawane były sygnały z wejścia licznika. Dla podanych wartości elementów częstotliwość generatora wynosi ok. 2 Hz, co pozwala na szybkie ustawienie żądanej cyfry.

Opracował
A.J.P.

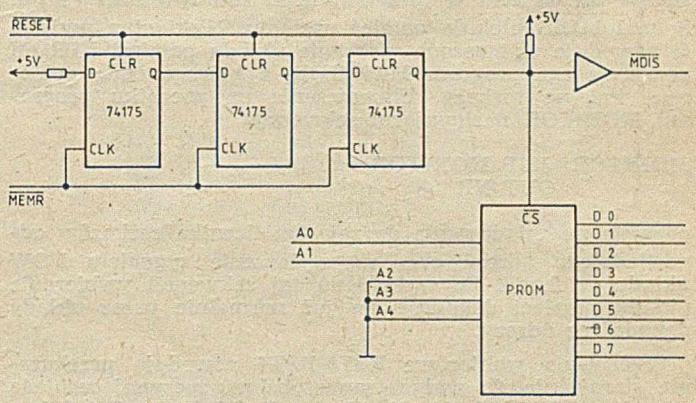
Oto kolejny przykład pokazujący, że stosowanie przestarzałej techniki zmusza do myślenia. Projektanci 16-bitowych mikroprocesorów wyeliminowali opisywany niżej problem, tak więc do dalszej lektury zapraszamy tych uparciuchów, którzy w mikrokomputerach stosują 8080.

W większości mikroprocesorów 8-bitowych podanie sygnału RESET powoduje wyzerowanie licznika rozkazów. Tym samym program powinien zaczynać się od adresu 0000H. Niby wszystko jest logiczne, jeśli przyjmiemy, że procesor wykonuje zawsze ten sam program zawarty w ROM-ie. Jeżeli jednak mamy do czynienia nie tyle z mikroprocesorowym sterownikiem, co z komputerem realizującym rozmaite programy wpisywane do RAM-u, to rozwiązanie logiczne okazuje się szalenie niewygodne (przerwania, umieszczanie tabel skoków, kompatybilność programów realizowanych na różnych systemach, itp). Stąd też powstał pomysł „oszukania" mikroprocesora, tak aby po sygnale RESET automatycznie przechodził do realizacji programu zarządzającego (zawartego w ROM-ie), umieszczonego pod koniec obszaru adresowego. Sposobów „oszukiwania" jest niemal tyle, ile systemów mikrokomputerowych. Są to jednak z reguły modyfikacje kilku podstawowych pomysłów. Dwa z nich przedstawiamy poniżej. Dla tych, którzy lubią fikuśne rozwiązania, warto dodać, że przedstawione pomysły można również wykorzystać dla obsługi przerwań wektorowych — generujących kody RST0...RST7.

Przekazywanie sterowania do programu MONITOR

Automatyczne przekazanie sterowania do programu zarządzającego po wystąpieniu sygnału RESET można zrealizować stosując proste rozwiązania sprzętowe.

Jako pierwsze przedstawiono rozwiązanie wykorzystujące dodatkową pamięć PROM (32 × 8 bajtów). W trzech początkowych komórkach pamięci zapisany jest rozkaz skoku do MONITORA (rys. 1). Sygnał RESET zeruje wyjścia przerzutników, co blokuje pamięć systemu (sygnałem $\overline{MDIS}$) oraz uaktywnia układ pamięci PROM. Trzy kolejne

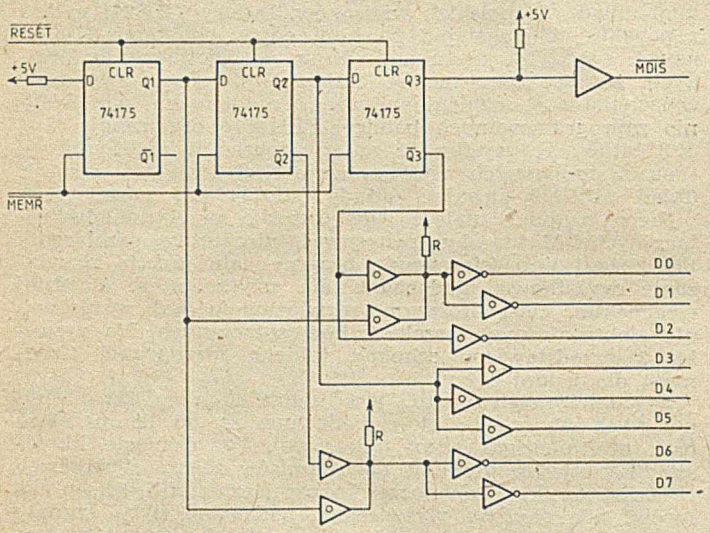


Rys. 1. Układ dostępu do programu zarządzającego po sygnale RESET z wykorzystaniem pamięci PROM

sygnały odczytu ($\overline{DBIN}$ lub $\overline{MEMR}$) powodują wprowadzenie i wykonanie przez procesor rozkazu skoku na początek programu MONITOR. Narastające zbocze trzeciego sygnału odczytu odblokowuje pamięć systemową. Pamięć PROM staje się nieaktywna do momentu podania kolejnego sygnału RESET. Wadą tego rozwiązania jest konieczność stosowania dodatkowej pamięci PROM. Układ można zmodyfikować wykorzystując bramki typu „otwarty kolektor" zamiast pamięci PROM.

Na rysunku 2 przedstawiono zmodyfikowaną wersję rozwiązania. W przykładzie założono, że program MONITOR rozpoczyna się od adresu F800H. W tabeli zilustrowano sposób generowania odpowiednich kodów na szynie danych. Wadą tego rozwiązania jest konieczność zmiany konfiguracji bramek dla różnych adresów początkowych programu MONITOR.

Całkowicie odmienne rozwiązanie przedstawione zostało na rysunku 3. Zastosowano tutaj „układ relokacji". Cechą charakterystyczną tego rozwiązania jest dekodowanie adresów programu zarządzającego w dwóch obszarach. Po



Rys. 2. Układ dostępu do programu zarządzającego po sygnale RESET z wykorzystaniem bramek otwarty kolektor

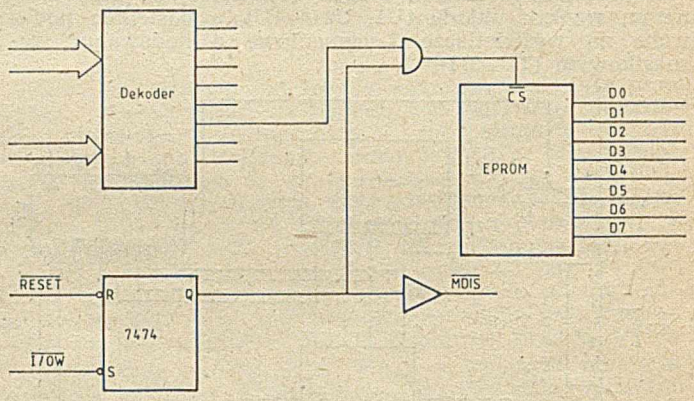
sygnale RESET jest to obszar od adresu 0000H, a po dokonaniu relokacji — obszar, w którym umieszczony jest program MONITOR. W odróżnieniu od poprzednich rozwiązań na początku programu MONITOR umieszczony jest skok do następnego rozkazu programu. Sygnał RESET ustawia na wyjściu Q (przerzutnik D lub RS) stan niski. Sygnał MDIS blokuje pamięć systemu. Niski stan wyjścia Q powoduje wymuszone wybranie układu pamięci EPROM zawierającego program MONITOR, z którego będą pobierane kolejne rozkazy. Umieszczenie na początku programu MONITOR następującej sekwencji:

```
MONITOR: JMP MONITOR + 3
         OUT NR
```

pozwala na załadowanie do rejestru licznika rozkazów odpowiedniego adresu oraz na wyłączenie sygnałem I/OW układu relokacji. Po wykonaniu tej sekwencji wybieranie układu pamięci dokonuje się już normalnie przez układy dekodujące adresy.

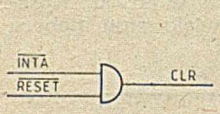
Przerwania wektorowe RST0-RST7 powodują przekazanie sterowania do podprogramu obsługi zaczynającego się od adresu 0, 8, 16, 24, 32, 40, 48, 56. Można uniknąć konieczności rezerwacji początku pamięci dla obsługi przerwań systemowych, wykorzystując w tym celu rozwiązania przedstawione na rysunkach 1 i 3.

Na wejścia zerujące przerzutników należy wprowadzić sygnał będący iloczynem logicznym sygnałów INTA i RESET (rys. 4). W pamięci PROM (64 bajty dla ośmiu przerwań) pod odpowiednimi adresami należy umieścić rozkazy skoków do podprogramów obsługi (rys. 5). Takie rozwiązanie pozwala na umieszczenie programów obsługi przerwań w dowolnym miejscu przestrzeni adresowej, umożliwiając swobodne dysponowanie początkiem pamięci.

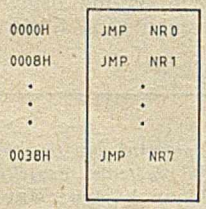


Rys. 3. Układ relokacji

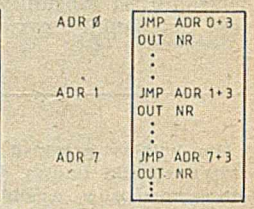
Podobnie można wykorzystać układ relokacji, w którym na wejście R przerzutnika należy wprowadzić iloczyn INTA i RESET. Pierwszymi rozkazami programu obsługi przerwania powinny być JMP i OUT (rys. 6).



Rys. 4. Sygnał zerowania przerzutników



Rys. 5. Obszar adresowy pamięci PROM



Rys. 6. Obszar adresowy podprogramów obsługi przerwań

LITERATURA

[1] Gates replace PROM in Intellec-8 bootstrap loader. Electronics, January 18, 1979, p. 121
[2] PROM adds bootstrap loader to Intellec-8 development system. Electronics, April 27, 1978, p. 126.

MAREK DORYWALSKI
PIE Warszawa

Sposób generowania sygnałów na szynie danych i otrzymywnia konfiguracji bramek

	JMP F800H	D7	D6	D5	D4	D3	D2	D1	D0	Q1	Q2	Q3	$\overline{Q1}$	$\overline{Q2}$	$\overline{Q3}$
RESET	C3	1	1	0	0	0	0	1	1	0	0	0	1	1	1
PO 1 $\overline{MEMR}$	00	0	0	0	0	0	0	0	0	1	0	0	0	1	1
PO 2 $\overline{MEMR}$	F8	1	1	1	1	1	0	0	0	1	1	0	0	0	1
PO 3 $\overline{MEMR}$	FF	1	1	1	1	1	1	1	1	1	1	1	0	0	0
				$\overline{Q1}+Q2$			Q2					Q3		$\overline{Q1}+Q3$	

Mikro-
-rewolucjoniści!
Do klawiatur!

Stare ekrany
czekają
na obraz!

Zaczynamy realizować zobowiązania. Oto pierwsze niebanalne SŁOWO. Pozwala ono uporządkować ciąg liczb według wielkości. Zdradzimy jednak, bez przypierania do muru, że jest to nadal bardziej program dydaktyczny niż użytkowy. Musimy jeszcze doskonalić znajomość FORTH'A. Niecierpliwym prymusom proponujemy — jako zadanie domowe — przeprowadzenie optymalizacji. Nadesłane prace zostaną ocenione przez ekspertów mikroKLANU!

FORTH (3)

Programowanie rozpoczniemy od zdefiniowania „własnego" stosu roboczego. Umieścimy go w pamięci powyżej słownika i bufora wejściowego. Wykorzystamy przy tym zmienną **HERE** określającą adres wierzchołka słownika. Określimy także stałą **REZERWA**, która zarezerwuje obszar 1 KB pamięci na bufor wejściowy. Zdefiniujemy również zmienną **HN** zawierającą adres wierzchołka stosu i dwa słowa pomocnicze:

H+ — przesunięcie wierzchołka stosu **H** w górę o 2 bajty i odłożenie aktualnego adresu na stos danych
H— — odłożenie adresu wierzchołka stosu **H** na stos danych i obniżenie wierzchołka stosu **H** o 2 bajty.

Definiowanie stosu (nazwiemy go stosem **H**) opisujemy trzema słowami głównymi:
HSET — inicjowanie stosu **H**
>H — przeniesienie liczby ze stosu danych na stos **H**
H> — przeniesienie liczby ze stosu **H** na stos danych.

Definiowanie stosu H

```
1024 CONSTANT REZERWA (definiowanie stałej REZERWA o wartości 1024)
0 VARIABLE HN (definicja zmiennej HN o wartości początkowej 0)

: HSET HERE @ REZERWA + (obliczenie początku stosu H)
      DUP HN ! (zapisanie w HN)
      0 SWAP ! (zapisanie wysokości stosu H równej 0)
;
: H+ HN@ (adres początku stosu H)
DUP@ 2 + (wysokość stosu H + 2)
OVER OVER SWAP ! (zapis nowej wysokości stosu H)
+ (adres wierzchołka stosu H)
;

      : H— HN@ (adres początku stosu H)
      DUP@ (wysokości stosu H)
       OVER OVER 2 — SWAP ! (zapis wysokości stosu H minus 2)
      +  (adres wierzchołka stosu H)
;
: >H H+ (adres na stosie H)
     !  (zapis)
: H> H— (adres na stosie H)
     @ (pobranie)
;
```

Przykład działania zdefiniowanych słów:
HSET 1 >H 2 >H 3 >H [CR] OK
H>.H>.H>.[CR] 3 2 1 OK

Definiowanie słowa SORT

Możemy teraz przystąpić do właściwego programu sortującego. W tym celu określimy cztery słowa pomocnicze:

MINIMUM — pobiera ze stosu danych liczbę określającą wysokość stosu i odszukuje wartość minimalną wśród liczb zapisanych na stosie; wyniki odkłada na stosie **H: (N1, N2, N3)**
N1 — adres elementu minimalnego na stosie danych (wierzchołek = 1)
N2 — zero
N3 — wartość elementu minimalnego
ZGUB — usuwa ze stosu danych liczbę wybraną przez **MINIMUM**, a ze stosu **H** usuwa liczby **N1** i **N2** zapisane przez **MINIMUM**
ZAPISZ — dokonuje rotacji trzech liczb ze stosu **H** i dekrementuje liczbę znajdującą się na wierzchołku stosu **H** oraz kopiuje wierzchołek stosu **H** na stos danych
ODDAJ — przenosi dane ze stosu **H** na stos danych; liczba z wierzchołka stosu **H** określa liczbę danych do przeniesienia.

```
: MINIMUM
          1 >H >H DUP >H (zainicjowanie wartości N1, N2, N3)
          BEGIN      (otwarcie bloku)
          H> DUP     (aktualna wartość minimalna)
          H> DUP >H 2 + (wysokość badanego stosu plus dwa parametry pobrane z H)
          PICK       (skopiowanie na wierzchołek stosu liczby leżącej na aktualnie „sondowanej" wysokości stosu)
          DUP ROT<   (porównanie z minimum)
          IF         (gdy mniejsze od minimum)
          >H >H >H DROP (zapisanie nowego minimum)
          ELSE       (gdy większe lub równe minimum)
          DROP >H    (zapis starego minimum)
          THEN       (koniec warunku IF)
          H> H>      (pobranie minimum i wysokości stosu)
          1 — SWAP OVER >H >H (zapis wysokości 1 i minimum)
          0 = (czy wysokość równa 0)
          UNTIL      (powtarzanie bloku BEGIN aż poprzedni warunek będzie spełniony)
;

: ZGUB
          H> H>      (pobranie N1, N2)
          DROP       (zgubienie N2)
          H>         (pobranie N3)
          SWAP >H    (zapis N1)
          ROLL       (rotacja stosu na wysokość N3)
          DROP       (zgubienie elementu wskazanego przez N3)
;

: ZAPISZ
          H> H> H>   (pobranie ze stosu H)
          ROT
          >H >H      (odłożenie na stos H)
          1 —        (zmniejszenie wartości o 1)
          DUP        (kopia dla stosu danych)
          >H         (zapis na stosie H)
;

: ODDAJ
          H>         (pobranie ilości liczb do przeniesienia)
          BEGIN      (otwarcie bloku)
                     H> SWAP (zapis na stosie danych)
                     1 — DUP (zmniejszenie ilości liczb do przeniesienia o 1)
                     0 = (koniec ? — ilość = 0)
          UNTIL      (powtarzanie bloku BEGIN aż poprzedni warunek będzie spełniony)
          DROP       (zgubienie zbędnego zera)
;

: SORT          (dydaktyczny program sortujący)
      DEPTH (odłożenie na stosie wysokości stosu)
       DUP >H >H (zapis wysokości sortowanego stosu)
```

```
BEGIN (otwarcie bloku)
      H> DUP >H MINIMUM (poszukiwanie minimum)
      ZGUB      (usunięcie znalezionego minimum)
      ZAPISZ    (zapamiętanie znalezionego minimum w odpowiednim miejscu stosu H i zmniejszenie wysokości badanego stosu o 1)
      0=        (wysokość badanego stosu = 0 ?)
      UNTIL     (powtarzanie bloku BEGIN aż poprzedni warunek będzie spełniony)
      H> DROP   (usunięcie zbędnego zera)
      ODDAJ     (przepisanie posortowanego stosu H na stos danych)
```

Przykład użycia programu

```
1 3 5 7 9 2 4 6 8 [CR] OK
HSET [CR] OK
SORT [CR] OK
..... [CR] 1 2 3 4 5 6 7 8 9 OK
```

Opisany program sortujący można łatwo zoptymalizować. Spróbuj!

MAREK CZARZYŃSKI
ABAKUS — Warszawa

TEST

Mikrokomputery bardzo trudno porównywać. Każdy ma swoje niejako dodatkowe możliwości, a także skrzętnie ukryte niedomagania. Pewnym ułatwieniem jest fakt, że prawie każdy może realizować programy napisane w języku BASIC. Ale — żeby życie nie było zbyt łatwe — nie ma chyba dwóch identycznych adaptacji tego języka. Projektanci lekką ręką dodają rozszerzenia i równie lekko wprowadzają ograniczenia. Rozpowszechniane są jednak programy wykorzystujące rdzeń języka, które pozwalają ocenić szybkość realizacji pewnych najczęściej stosowanych procedur. Programy takie (ang. Benchmark Program) często są pisane na zamówienie producentów, tak aby wykazać wyższość konkretnego komputera nad innymi. Zamieszczone poniżej procedury wydają się być stosunkowo obiektywne. Zaczerpnięte zostały z biuletynu Personal Programing Center — PPC Computer Journal V1N1P10 1982. Poniżej podajemy tabelę z wynikami testu dla siedmiu popularnych mikrokomputerów. Jeżeli nie ma wśród nich Twojego, będziemy wdzięczni za przesłanie nam uzupełnienia do tabeli.

Program 1

```
300 PRINT "START"
400 FOR K=1 TO 1000
500 NEXT K
700 PRINT "END"
800 END
```

Program 2

```
300 PRINT "START"
400 K=0
500 K=K+1
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
```

Program 3

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/K*K+K—K
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
```

Program 4

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/2*3+4—5
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
```

Program 5

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/2*3+4—5
520 GOSUB 820
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
820 RETURN
```

Program 6

```
300 PRINT "START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4—5
520 GOSUB 820
530 FOR L=1 TO 5
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
820 RETURN
```

Program 7

```
300 PRINT "START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4—5
520 GOSUB 820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
820 RETURN
```

Oprac.
J. TATARKIEWICZ

Tabela wyników testu

Komputer	Język	Czas wykonywania poszczególnych programów (s)							
		1	2	3	4	5	6	7	suma
OSBORNE I	MBASIC	1,5	4,5	12,1	11,8	12,8	23,0	36,2	101,9
APPLE II	APPLESOFT	1,4	8,3	15,7	17,5	19,1	28,4	44,5	134,9
HP-85	HP BASIC	2,5	4,5	17,1	17,2	18,4	30,4	45,5	135,6
HP-75	HP BASIC*)	3,1	5,1	21,6	21,3	26,3	41,9	59,4	178,7
ZX SPECTRUM	BASIC	4,3	8,14	20,02	19,26	23,06	53,12	77,48	205,42
TRS-80	BASIC II	2,8	11,3	27,7	29,5	32,8	54,5	82,5	241,1
SHARP PC-1500	BASIC II	14,8	29,5	78,9	79,5	94,0	168,9	223,4	689,0
Twój komputer	BASIC	...	...	...	...	...	...	...	...

*) zmienne typu REAL

Potrzebna pomoc

(pod tym hasłem zamieszczać będziemy anonse firm i osób prywatnych)

- Firma INSTAL zakupiła mikrokomputery IMP85. Poszukuje oprogramowania użytkowego pomocnego w prowadzeniu przedsiębiorstwa. Poszukiwana jest również realizacja podłączenia IMP85 do ODRY 1305 (pracującej pod systemem GEORGE-3) w charakterze inteligentnego terminala, mogącego wyeliminować dziurkarki kart. Zgłoszenia przyjmuje Jarosław Zrost; INSTAL—Łódź, ul. Brukowa 20, tel. 51-21-19 w. 285.
- Otrzymaliśmy informację od Jana Rabizo z Sosnowca o istnieniu Agencji Mikro-Komputerowej AMiCO, oferującej usługi obliczeniowe, programy mikrokomputerowe i doradztwo informatyczne. Programy można otrzymać w postaci wydruku lub zapisane na taśmie magnetycznej.
  Kontakt: Agencja AMICO, Sosnowiec, P-157, tel. 69-96-49 (godz. 18—20).
- Pisaliśmy już o możliwości wymieniania programów dla ZX SPECTRUM. Zbiór Piotra Parlewicza (tel. 42-40-94 w Warszawie) przekroczył liczbę 100!

- **INFORMATYKA tel. 27-71-40**
- **ABAKUS tel. 42-91-85**
- **KUM tel. 41-26-01**
- **?**

— prowadzi Andrzej J. Piotrowski (tel. dom. 48-22-85)